L7: Entry 2 of 2 File: USPT Jun 6, 2000

DOCUMENT-IDENTIFIER: US 6073124 A
TITLE: Method and system for securely incorporating electronic information into an online purchasing application

Brief Summary Text (5):
One of the major problems that authors of electronic content face using digital commerce is a reliable mechanism for obtaining payment for their electronic content. One reason is that it has become increasingly easy, without the use of secure licensing code, to copy and widely distribute electronic content. To limit the use of illegal copies of electronic content, current systems have incorporated licensing code into existing application programs to be electronically distributed using various solutions. According to one technique, which will be referred to herein as "wrapping," a second application program (a wrapper program) is distributed on the network, which includes an encrypted version of the original application program. The wrapper program, when installed, decrypts the encrypted original application program and then proceeds to execute the original application program. To successfully decrypt the program, a legitimate end user must provide the proper licensing information to enable the decryption to operate. A security hole exists, however, in that, while the wrapping program is in the process of decrypting the original application executable file, temporary files are created to hold the decrypted program code. Once the entire original application program has been decrypted and stored in the temporary file, a "software pirate" can then make multiple copies of the original unencrypted application program in the temporary file and can distribute them illegally.

Brief Summary Text (6):
Further, use of the wrapping technique to incorporate licensing provides only limited additional security to a vendor who implements what is known as a "try and buy" licensing model. A try and buy licensing model typically distributes an application program with either limited functionality or for a limited time of use to enable a potential customer to explore the application. Functionality may be limited, for example, by disabling a set of features. Once the potential customer is satisfied, the customer can pay for and license the application program for more permanent use. If an application program is distributed using the wrapping technique to potential customers for the purpose of try and buy licensing, then, when the application program is decrypted and stored in a temporary file, a software pirate can determine how to enable the disabled features or how to remove the license expiration data. These security problems can result in the distribution of illegal copies, which are hard to detect and monitor in a global network environment.

Detailed Description Text (6):
The secure digital commerce system is arranged according to a client/server architecture and provides a modularized DCS client and a modularized DCS server that interact with the online purchasing system to perform a purchase. The DCS client includes a set of client components; support for downloading the client components onto a customer computer system; and support for communicating with the DCS server to license an item of merchandise. The client components contain a

secured (e.g., encrypted) copy of the content and various components needed to license and purchase the merchandise and to unsecure (e.g., decrypt) and execute the licensed merchandise. The DCS client communicates with the DCS server to download the client components onto a customer's computer system in response to a request for merchandise from the online purchasing system. The DCS client also communicates with the DCS server to license and purchase the requested merchandise. The DCS server generates an electronic license certificate, which contains license parameters (e.g., terms) that are specific to the requested merchandise and to a desired purchasing option (such as trial use, permanent purchase, or rental). The DCS server then sends the generated electronic license certificate to the DCS client. Once a valid electronic license certificate for the requested merchandise is received by the DCS client, the merchandise is made available to the customer for use in accordance with the license parameters contained in the electronic license certificate.

Detailed Description Text (15):
FIG. 4 is an overview flowchart of the example steps performed by the secure digital commerce system components to perform the licensing and purchase of electronic data. This figure briefly describes the interactions between the components shown in FIG. 3 to accomplish the downloading, licensing, and purchasing of a requested item of merchandise when it can be delivered online. In step 401, the potential customer downloads a WEB page (part of the customer front end 312) from the virtual store 304 that includes the item to be requested (see, for example, FIG. 2). In step 402, the customer requests an item of merchandise, for example, by selecting an icon that is linked to a download file that corresponds to the desired item. In response to the selection, in step 403, the virtual store 304 downloads and installs the download file, which extracts the executable boot program and component list and causes execution (preferably as a background task) of the executable boot program on the customer computer system 311. In step 404, the boot program reads the component list to determine what DCS client components to download and requests the determined components from the appropriate contents supplier server 306. The component list, as further described below with reference to Table 2, indicates source and target locations for each component to be downloaded. In step 405, the boot program installs a downloaded (secured) content file that is associated with the desired item of merchandise and causes the content file to be processed (executed). When the content file is a computer program, then the downloaded content file has been previously configured to automatically cause licensing code to be executed before the content file is executed. When instead the content file is data to be input to a computer program, then the content player is previously configured to automatically cause the licensing code to be executed first before the content file data is processed. More specifically, the downloaded content player is installed by the boot program to process the secured (e.g., encrypted) content file data. The boot program then starts the execution of the content player, which invokes and causes execution of the downloaded licensing code. Thus, in step 406, the licensing code, which is incorporated into either the content file or the content player, is executed. In step 407, if the licensing code determines that a valid ELC already exists, then the content file continues to be processed in step 412, else the licensing code continues in step 408. In step 408, the licensing code requests a valid ELC from the licensing and purchasing broker 307. In step 409, the licensing and purchasing broker 307 determines whether a purchase is requested and, if so, continues in step 410, else continues in step 411. In step 410, the licensing and purchasing broker 307 obtains a method for payment and authorizes the payment method using the payment processing function 309. In step 411, the licensing and purchasing broker 307 generates an appropriate ELC using the licensing library 310 and the password generation data repository 308 and returns the generated EL-C to the licensing code. In step 412, if portions of the content file are encrypted as will be further described, then the content file is decrypted and processed.

Detailed Description Text (16):

h e b b g e e e f c e e g f e ge

As indicated above, when the downloaded (secured) content file is a computer program, licensing code is automatically invoked to verify the existence of, or obtain, a valid electronic license certificate for a requested item and to decrypt and execute the content file. One mechanism for incorporating licensing code into a content file such that it is automatically invoked is discussed in detail with reference to related U.S. patent application Ser. No. 08/792,719, entitled "Method and System for Injecting New Code Into Existing Application Code," filed on Jan. 29, 1997. That patent application describes a technique for inserting licensing code into an existing application and for inserted security code that securely executes the application code. The security code uses an incremental decryption process to ensure that a complete version of the unmodified application code is never visible at any one time (to avoid illegitimate copying). Thus the security code mechanism described therein makes it impossible for someone to create an unmodified version of the application in a reasonable amount of time. The insertion technique described therein can be used to insert into a content file the licensing code component of the DCS client, which communicates with the licensing and purchasing broker to generate an ELC. Further, the encryption/decryption technique described therein may be used in the current context to incorporate security code that securely decrypts and executes the downloaded content file.

Detailed Description Text (17):
In addition, when the content file is data to be used as input to a computer program (such as a content player), then the licensing code can be incorporated into the computer program by invoking licensing code and security code routines. For example, an application programming interface ("API") to the licensing code and to the incremental decryption security code can be provided. The content player is programmed (or configured via the insertion technique described in the related patent application) to include calls to the API routines to validate or obtain an ELC and to unsecure (e.g., decrypt) the associated content file. One skilled in the art will recognize that any mechanism that automatically causes the execution of licensing code (and security code) before the secured content is processed is operable with embodiments of the present invention.

Detailed Description Text (21):
Specifically, in step 601, the utility incorporates licensing and security code into the supplier specific electronic content or content player. As described above, an exemplary embodiment incorporates licensing and security code according to the techniques described in the related U.S. patent application Ser. No. 08/792,719, entitled "Method and System for Injecting New Code into Existing Application Code," filed on Jan. 29, 1997 or by calling routines of an API as appropriate (e.g., when a content player is needed). One skilled in the art, however, will recognize that any technique for ensuring that proper licensing code gets executed when the content is processed and for encrypting (and subsequently decrypting) the content file will operate with embodiments of the present invention. In step 602, the utility produces one or more files that contain the (partially or fully) encrypted content. In step 603, the utility produces an encrypted DCS security information file(s), which contain information that is used, for example, to decrypt the content and to produce a proper license. The contents of an encrypted DCS security information file are described in further detail below with reference to Table 1. In step 604, the utility creates a component list file (an ".ssc" file) and a download file for this particular online merchandise. Specifically, in an embodiment that statically generates download files, a self-extracting installation file is generated (the download file), which contains the component list file (an ".ssc" file) specific to the merchandise and the executable boot program. As described above, the download file, which contains the executable boot program and the component list, is typically stored on the virtual store computer system. The executable boot program uses the component list file to determine the components to download and to download them when particular electronic content is requested. An example component list file is described further below with reference to Table 2. In step 605, the utility stores the

download file on the virtual store computer system (e.g., virtual store 304 in FIG. 3). When instead the download files are dynamically generated by the virtual store when needed for a particular WEB page, then in steps 604 and 605, the utility creates and stores only the component list file. In step 606, the utility stores the other components of the DCS client, for example, the encrypted content and DCS security information files, the licensing code, and the user interface library on the content supplier server system (e.g., content supplier server 306 in FIG. 3). In step 607, the utility updates the password generation data repository (e.g., password generation database 308 in FIG. 3) with the merchandise-specific licensing information, for example, the fields used to generate the license parameters of a valid electronic license certificate, and then returns. An example password generation data repository is discussed in further detail with reference to Tables 3, 4, and 5. One skilled in the art will recognize that the generation of these components and the password generation data may be performed at different times and by separate utilities.

Detailed Description Text (23):
data in the encrypted DCS security information file is encrypted separately from the content file to enable multiple items of merchandise to share purchasing, licensing, and decryption information. This capability is especially useful when the items are provided by the same content supplier server. Thus, a single encrypted DCS security information file may be associated with more than one encrypted content file. In addition, each field in the DCS security information file is encrypted separately. By separately encrypting each field, purchasing or licensing information can be changed without having to re-encrypt the content file or the rest of the DCS security information file.

Detailed Description Text (24):
Specifically, in Table 1 the CommerceServer field indicates the location of the licensing and purchasing broker (e.g., the network address of licensing and purchasing broker 307 in FIG. 3) to be used to license and purchase the merchandise. (In embodiments of the secure digital commerce system, one or more content suppliers, licensing and purchasing brokers, or payment processing functions, may be utilized.) The ProductSKUId field is a specific identifier associated with a version (each executable) of a product for a specific reseller (virtual store). For the purposes of example, exemplary embodiments assume that a product may have multiple versions and that each version may be packaged differently depending upon the purchasing option (for example, trial use versus full purchase). In addition, more than one reseller may offer a version of a product. The ProductSKUId field is used to identify a password configuration table to be used to generate an electronic license certificate and is discussed further below. The ProductUUID field is a specific identifier associated with each version of a product regardless of the reseller. By using an identifier that is specific to the product version and not to the reseller, the digital commerce system can ensure that a customer who licenses a version of a product for (one time) trial use may not utilize multiple resellers to obtain more than one ELC for the same version. In addition, this identifier is used by the licensing code to locate the associated DCS security information file and is associated with various licensing-specific information. For example, clock data can be stored in a system registry indexed by ProductUUID to ensure that "time-bomb" protected content is not defeated by resetting the clock to illegitimately process the content. The UILibName indicates the location of a user interface library to be used for purchasing the merchandise. The EntryPoint, ImageBase, EKey, ECode, DataSize, NumberRelocations, and Relocations fields are used to support the decryption of the encrypted content file (s) and to determine the relocation information when the content file is secured using the technology of related U.S. patent application Ser. No. 08/792,719. If an alternative licensing and encryption scheme is used, then these fields would be modified accordingly. The ContactCompany, ContactAddress, ContactSupportPhone, ContactSupportFax, ContactSupportEmail, ContactOrderPhone, ContactOrderFax, and ContactOrderEmail fields reflect supplier dependent information that can be

displayed in dialogs presented by the virtual store depending on the user interface being employed. The DeveloperID and SecretKey fields are used to create a symmetric key to decode the electronic license certificate generated by the licensing and purchasing broker. The other fields are used for other similar licensing and purchasing functions.

Detailed Description Text (31):
One skilled in the art will recognize that different behaviors will occur when the content file (or content player) begins executing depending upon the technique used to incorporate the licensing code and decryption (security) code and depending upon the encryption/decryption technique used. For example, as described in further detail in related U.S. patent application Ser. No. 08/792,719, when using the injection techniques described therein, the execution of the encrypted content file will automatically cause the licensing code and (eventually) the security code to be executed as a result of injecting a licensing DLL into the content file. Specifically, a "DLLMain" routine is automatically invoked when the licensing code library is loaded, which in turn executes the actual licensing code. After the licensing code executes, the security code stored in the encrypted content automatically executes because it is inserted into the content file immediately following (a reference to) the licensing code. Thus, the licensing code and the decryption code are automatically executed before any supplier-specific content is executed. The security code in an exemplary embodiment decrypts the encrypted content incrementally in order to prevent a fully decrypted version of the content to be present in its entirety at any one time. A similar procedure is used when the content player invokes the licensing and security code with an exception that the licensing and security code is explicitly invoked and knows how to locate the content file and to decrypt it incrementally.

Detailed Description Text (34):
broker. For example, other requests associated with rental use or other types of purchasing options may be supported. The processing of these HTTP request messages by the licensing and purchasing broker is discussed further below with respect to FIG. 12. In step 908, the licensing code receives a valid ELC from the licensing and purchasing broker, stores it, and continues in step 909. The ELC may be stored in any area that is accessible to processes executing on the customer computer system, such as in a system registry. In step 909, the licensing code causes the decryption and execution of the licensed content, and returns.

Detailed Description Text (37):
FIG. 11 is an example flow diagram of the steps performed by licensing code to determine whether a valid electronic licensing certificate is available. In step 1101, the code retrieves, decrypts, and decodes the electronic licensing certificate (ELC) to obtain the parameters of the license (e.g., the license terms). The license parameters that are obtained in step 1101 indicate, for example, how many uses of a particular license can be executed or, for example, how many different user passwords are able to use the same electronic license. In addition, license parameters that reflect an authorized time period for use may be specified. In step 1102, the code tests various attributes of the customer computer system to determine whether the conditions indicated by the retrieved license parameters have been met. In step 1103, if all of the conditions have been met (for example, the license use period has not expired), then the code returns indicating that a valid license is in effect. Otherwise, the code returns indicating that the current license is invalid.

Detailed Description Text (38):
In an exemplary embodiment, the ELC is encrypted and decrypted using a symmetric key algorithm. A symmetric algorithm implies that the same key is used to encrypt a plaintext message and to decrypt a ciphertext message. Any symmetric key algorithm could be used. Symmetric and public key cryptography, both of which are utilized by exemplary embodiments of the present invention, are described in detail in Bruce

Schneier, Applied Cryptography, John Wiley & Sons, Inc., 1994, which is herein incorporated by reference. According to one technique, the DeveloperID and SecretKey fields (stored in the encrypted information file) are used to formulate a symmetric key, which is client and product specific. These fields are provided by the supplier when the SAFEmaker utility is executed to produce the components of the DCS client (see FIG. 6). Because the encryption of the ELC is provided by the licensing and purchasing broker and the corresponding decryption of the ELC is provided by the licensing code, the encryption and decryption code are preferably synchronized to correspond to one another. For this reason, a separate dynamic link library (e.g., passgen.dll) is used by the licensing and purchasing broker to allow the encryption algorithm to be replaced at any time to correspond to different licensing code.

Detailed Description Text (41):
In step 1206, the broker determines whether it has received an HTTP request message that indicates trial use is desired and, if so, continues in step 1207, else continues in step 1209. In step 1207, in order for the broker to generate an ELC specific to the user and to the indicated product, certain information is typically sent by the licensing code in the HTTP request message. Specifically, information that uniquely identifies the user and the product version is provided. The broker uses the received product version identifier (the ProductSKUId) to retrieve from a version table a corresponding password configuration identifier (pass-config-id). Once the pass-config-id is retrieved from the version password generation data repository table, this identifier is used as an index into a password configuration table to determine a set of fields to be used to generate the license parameters of the ELC. (One will recall that the fields stored in the password generation tables were specified by the supplier of the content in conjunction with the SAFEmaker utility.) An example password configuration table is shown below as Table 3. A table with potentially different fields exists for each unique pass-config-id. Because multiple versions of products and multiple products may use the same pass-config-id, they may share a single password configuration table. This attribute may be useful, for example, if all the products from a particular supplier have similar electronic licensing capabilities. In step 1208, an ELC is generated based upon the fields of the determined password configuration table using a symmetric key formulated from the SecretKey and DeveloperID fields of the encrypted information file and an appropriate encryption algorithm, as discussed earlier. For the purposes of this specification, the ELC may be viewed as a very long number which encrypts the license parameters indicated by the fields in the password configuration table. In an exemplary embodiment, the code used to perform steps 1207-1208 is provided in a separate code module (e.g., passgen.dll) so that the password generation code, including the encryption and decryption algorithms, can be easily replaced in a licensing and purchasing broker.

Detailed Description Text (45):
Communications between the DCS client components and the licensing and purchasing broker are preferably performed using a secure communication methodology. FIG. 19 is an example block diagram that illustrates one technique for ensuring secure communication between a DCS client component and a licensing and purchasing broker. Although FIG. 3 may imply that the downloaded components communicate with the licensing and purchasing broker to request licensing and purchasing and to receive the generated ELC, one skilled in the art will recognize that it is also possible for these components to communicate via a server associated with the virtual store. In FIG. 19, communication between the client components (clients) 1901 and 1902 and the licensing and purchasing broker 1903 depends upon secure key exchange. Secure key exchange is accomplished by sending a client-specific symmetric key using a public/private key algorithm. The client-specific symmetric key is used solely for communication between that client and the licensing and purchasing broker. Specifically, a separate communication session-specific symmetric key is provided by each client for each communication session and is sent to the licensing and purchasing broker 1903 in a session initiation message using the broker's public

key. One technique for distributing and obtaining the broker's public key is to use a commercially available digital signature service, such as Verisign. Because the broker 1903 is the only process that knows its own private key, the broker 1903 decrypts the session initiation message using its private key and retrieves the client's session-specific symmetric key. Thereafter, all messages from the broker 1903 to the client 1901 are encrypted by the broker 1903 using the client 1901's symmetric key. Client 1901 is then able to decrypt a received message using the symmetric key that it initially generated and sent to the broker 1903. Client 1901 encrypts messages to send to the broker 1903 also using client 1901's symmetric key. Similarly, the client 1902 sends its own encrypted symmetric key to broker 1903 using the broker's public key. The broker 1903 in turn communicates with the client 1902 using the client-specific symmetric key that corresponds to client 1902.

Detailed Description Text (49):
Table 3 is an example password configuration table. As described earlier, a separate password configuration table is provided for each password configuration identifier (pass-config-id). There is a version table in the data repository for translating between a retailer specific product version identifier (the ProductSKUId) and a corresponding password configuration identifier. The fields are used to generate the license parameters for an ELC that corresponds to the determined password configuration identifier. One skilled in the art will recognize that any fields could be stored in the password configuration table. Further, any algorithm for combining the fields in a determinable fashion to encrypt them into a single code that can be decrypted without losing information could be utilized to generate the ELC.

Previous Doc Next Doc Go to Doc#

L9: Entry 2 of 2 File: USPT Jun 6, 2000

DOCUMENT-IDENTIFIER: US 6073124 A
TITLE: Method and system for securely incorporating electronic information into an online purchasing application

Brief Summary Text (9):
To perform digital commerce, today's computer networking environments utilize a client/server architecture and a standard protocol for communicating between various network sites. One such network, the World Wide WEB network, which comprises a subset of Internet sites, supports a standard protocol for requesting and for receiving documents known as WEB pages. This protocol is known as the Hypertext Transfer Protocol, or "HTTP." HTTP defines a high-level message passing protocol for sending and receiving packets of information between diverse applications. Details of HTTP can be found in various documents including T. Berners-Lee et al., Hypertext Transfer Protocol--HTTP 1.0, Request for Comments (RFC) 1945, MIT/LCS, May, 1996, which is incorporated herein by reference. Each HTTP message follows a specific layout, which includes among other information a header, which contains information specific to the request or response. Further, each HTTP message that is a request (an HTTP request message) contains a universal resource identifier (a "URI"), which specifics a target network resource for the request. A URI is either a Uniform Resource Locator ("URL") or Uniform Resource Name ("URN"), or any other formatted string that identifies a network resource. The URI contained in a request message, in effect, identifies the destination machine for a message. URIs, as an example of URIs, are discussed in detail in T. Berners-Lee, et al., Uniform Resource Locators (URL), RFC 1738, CERN, Xerox PARC, Univ. of Minn., December, 1994, which is incorporated herein by reference.

Brief Summary Text (11):
In FIG. 1, a WEB browser application 101 is shown executing on a client computer system 102, which communicates with a server computer system 103 by sending and receiving HTTP packets (messages). The WEB browser application 101 requests WEB pages from other locations on the network to browse (display) what is available at these locations. This process is known as "navigating" to sites on the WEB network. In particular, when the WEB browser application 101 "navigates" to a new location, it requests a new page from the new location (e.g., server computer system 103) by sending an HTTP-request message 104 using any well-known underlying communications wire protocol. HTTP-request message 104 follows the specific layout discussed above, which includes a header 105 and a URI field 106, which specifies the target network location for the request. When the server computer system machine specified by URI 106 (e.g., the server computer system 103) receives the HTTP-request message, it decomposes the message packet and processes the request. When appropriate, the server computer system constructs a return message packet to send to the source location that originated the message (e.g., the client computer system 102) in the form of an HTTP-response message 107. In addition to the standard features of an HTTP message, such as the header 108, the HTTP-response message 107 contains the requested WEB page 109. When the HTTP-response message 107 reaches the client computer system 102, the WEB browser application 101 extracts the WEB page 109 from the message, and parses and interprets the HTML code in the page (executes the WEB page) in order to display the document on a display screen

of the client computer system 102 in accordance with the HTML tags.

Detailed Description Text (4):
Each icon is typically linked to a server site on the network, which is responsible for supplying the content of the item when purchased if the item is capable of electronic delivery. When the user selects one of the icons, the browser application, as a result of processing the link, sends a request for the selected item to the server site. Thus, when a customer selects the icon 203, an HTTP request message is sent to an appropriate server site to locate and download the software modules that correspond to "RETURN OF ARCADE."

Detailed Description Text (11):
However, in an exemplary embodiment, the secure digital commerce system utilizes the HTTP request communication model provided by the World Wide WEB network. A detailed description of this architecture and of WEB page communication is provided in J. O'Donnell et al., Special Edition Using Microsoft Internet Explorer 3, QUE Corp., 1996, which is incorporated herein by reference.

Detailed Description Text (33):
Specifically, in step 901, the licensing code determines whether a valid electronic license certificate ("ELC") is available. The steps used to make this determination are discussed further below with reference to FIG. 11. If a valid ELC is available, then the licensing code continues in step 909 and skips the licensing and purchasing process, else continues in step 902. In step 902, the licensing code loads the user interface library associated with the component and obtains a purchase option from the customer, such as "rent-to-buy," "buy," or "try." The purchase options assist in determining the parameters of a valid license. An example interface for obtaining this information is described below with reference to FIG. 10. The licensing code obtains the user interface library name by retrieving the UILibName field from the DCS security information file associated with the product. The associated DCS security information file can be determined from the ProductUUID, which was previously stored in the system registry by the boot program during the component download process. In step 903, the licensing code determines whether the customer has indicated that a trial purchasing option is requested and, if so, continues in step 904, else continues in step 905. In step 904, the licensing code sends an HTTP request message to the licensing and purchasing broker (e.g., the licensing and purchasing broker 307 in FIG. 3) to provide an appropriate license for trial use of the product, and continues in step 908. In step 905, the licensing code determines whether the customer has indicated a purchasing option to purchase the content and, if so, continues in step 906, else continues in step 907. In step 906, the licensing code sends an HTTP request message to the licensing and purchasing broker to purchase the content, and continues in step 908. In step 907, the licensing code determines whether any other type of licensing or purchasing request has been indicated by the customer and sends an appropriate HTTP request message to the licensing and purchasing

Detailed Description Text (34):
broker. For example, other requests associated with rental use or other types of purchasing options may be supported. The processing of these HTTP request messages by the licensing and purchasing broker is discussed further below with respect to FIG. 12. In step 908, the licensing code receives a valid ELC from the licensing and purchasing broker, stores it, and continues in step 909. The ELC may be stored in any area that is accessible to processes executing on the customer computer system, such as in a system registry. In step 909, the licensing code causes the decryption and execution of the licensed content, and returns.

Detailed Description Text (39):
FIG. 12 is an example flow diagram of the steps performed by a licensing and purchasing broker of the secure digital commerce system. These steps are executed in response to receiving an HTTP request message sent by the licensing code in step

904 or 906 in FIG. 9. As described earlier, the licensing and purchasing broker interacts with a password generation system (e.g., passgen.dll and the data repository) and payment processing functions to license and purchase an indicated item of merchandise. In summary, when the licensing and purchasing broker receives a request to buy an item, it performs appropriate payment processing to perform a purchase. When the licensing and purchasing broker receives either a request to try or a request to buy the item, the broker uses the password generation system to generate an ELC to return to the licensing code.

Detailed Description Text (41):
In step 1206, the broker determines whether it has received an HTTP request message that indicates trial use is desired and, if so, continues in step 1207, else continues in step 1209. In step 1207, in order for the broker to generate an ELC specific to the user and to the indicated product, certain information is typically sent by the licensing code in the HTTP request message. Specifically, information that uniquely identifies the user and the product version is provided. The broker uses the received product version identifier (the ProductSKUId) to retrieve from a version table a corresponding password configuration identifier (pass-config-id). Once the pass-config-id is retrieved from the version password generation data repository table, this identifier is used as an index into a password configuration table to determine a set of fields to be used to generate the license parameters of the ELC. (One will recall that the fields stored in the password generation tables were specified by the supplier of the content in conjunction with the SAFEmaker utility.) An example password configuration table is shown below as Table 3. A table with potentially different fields exists for each unique pass-config-id. Because multiple versions of products and multiple products may use the same pass-config-id, they may share a single password configuration table. This attribute may be useful, for example, if all the products from a particular supplier have similar electronic licensing capabilities. In step 1208, an ELC is generated based upon the fields of the determined password configuration table using a symmetric key formulated from the SecretKey and DeveloperID fields of the encrypted information file and an appropriate encryption algorithm, as discussed earlier. For the purposes of this specification, the ELC may be viewed as a very long number which encrypts the license parameters indicated by the fields in the password configuration table. In an exemplary embodiment, the code used to perform steps 1207-1208 is provided in a separate code module (e.g., passgen.dll) so that the password generation code, including the encryption and decryption algorithms, can be easily replaced in a licensing and purchasing broker.

Detailed Description Text (55):
FIG. 21 is an example flow diagram of the additional steps performed by a licensing and purchasing broker of the secure digital commerce system to support non-ESD transactions. In step 2102, the licensing and purchasing broker selects the next item of merchandise requested starting with the first. FIG. 21 assumes that each HTTP request may request more than one item of merchandise. For example, a user interface library may offer additional non-ESD merchandise, which can be purchased at the same time that a customer purchases an ESD item. The user interface library generates and sends to the licensing and purchasing broker an HTTP request, which requests the purchase of multiple items of merchandise. For each item in the purchase request, in steps 2103-2110, the broker processes the item in accordance with an indicated purchasing option for the item.

Detailed Description Text (59):
Although specific embodiments of, and examples for, the present invention are described herein for illustrative purposes, it is not intended that the invention be limited to these embodiments. Equivalent methods, structures, processes, steps, and other modifications within the spirit of the invention fall within the scope of the invention. For example, the teachings provided herein of the present invention can be applied to other client/server architectures, not necessarily the exemplary Internet based, HTTP model described above. These and other changes may be made to

the invention in light of the above detailed description. Accordingly, the invention is not limited by the disclosure, but instead the scope of the present invention is to be determined by the following claims.

Detailed Description Paragraph Table (2):

TABLE 2

<Execute TRIGGER = "<ProgramFilesDir>.backslash.winzip.backslash.winzip32.exe" URI = "http://devserver/products/winzip32/winzipsetup.exe" MSGDIG = "NDLsrKcS36YbugITP4yUjv8PSfk=" ProductUUID = "WINZIP-demo-0000" NAME = "WinZip 6.2" DESCRIPTION = "WinZip 6.2" LOCAL = "<ProgramFilesDir>.backslash.winzip.backslash.setup.exe">

Other Reference Publication (1):

T. Berners-Lee et al., "Hypertext Transfer Protocol--HTTP 1.0," Request for Comments (RFC) 1945, MIT/LCS, May, 1996.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

Generate Collection Print

L10: Entry 2 of 2 File: USPT Jun 6, 2000

283

DOCUMENT-IDENTIFIER: US 6073124 A
TITLE: Method and system for securely incorporating electronic information into an online purchasing application

Brief Summary Text (10):
FIG. 1 illustrates how a browser application, using the client/server model of the World Wide WEB network, enables users to navigate among network nodes by requesting and receiving WEB pages. For the purposes of this specification, a WEB page is any type of document that abides by the HTML format. That is, the document includes an "<HTML>" statement. Thus, a WEB page can also be referred to as an HTML document or an HTML page. HTML is a document mark-up language, defined by the Hypertext Markup Language ("HTML") specification. HTML defines tags for specifying how to interpret the text and images stored in an HTML page. For example, there are HTML tags for defining paragraph formats and text attributes such as boldface and underlining. In addition, the HTML format defines tags for adding images to documents and for formatting and aligning text with respect to images. HTML tags appear between angle brackets, for example, <HTML>. Further details of HTML arc discussed in T. Berners-Lee and D. Connolly, Hypertext Markup Language-2.0, RFC 1866, MIT/W3C, November, 1995, which is incorporated herein by reference.

Brief Summary Text (11):
In FIG. 1, a WEB browser application 101 is shown executing on a client computer system 102, which communicates with a server computer system 103 by sending and receiving HTTP packets (messages). The WEB browser application 101 requests WEB pages from other locations on the network to browse (display) what is available at these locations. This process is known as "navigating" to sites on the WEB network. In particular, when the WEB browser application 101 "navigates" to a new location, it requests a new page from the new location (e.g., server computer system 103) by sending an HTTP-request message 104 using any well-known underlying communications wire protocol. HTTP-request message 104 follows the specific layout discussed above, which includes a header 105 and a URI field 106, which specifies the target network location for the request. When the server computer system machine specified by URI 106 (e.g., the server computer system 103) receives the HTTP-request message, it decomposes the message packet and processes the request. When appropriate, the server computer system constructs a return message packet to send to the source location that originated the message (e.g., the client computer system 102) in the form of an HTTP-response message 107. In addition to the standard features of an HTTP message, such as the header 108, the HTTP-response message 107 contains the requested WEB page 109. When the HTTP-response message 107 reaches the client computer system 102, the WEB browser application 101 extracts the WEB page 109 from the message, and parses and interprets the HTML code in the page (executes the WEB page) in order to display the document on a display screen of the client computer system 102 in accordance with the HTML tags.

Previous Doc Next Doc Go to Doc#